

Formal Method to Analyze Design Workflows in Software Development

N. N. Voit(✉) and S. I. Bochkov(✉)

Ulyanovsk State Technical University, Ulyanovsk, Russia
n.voit@ulstu.ru

Abstract. In this paper we present a new formal method for the design workflows and software algorithms analysis in the form of algorithm graph-schemes based on the graphic languages, using the EPC language example. This method is based on the RVTI temporal automaton grammar, it also provides detection of temporal and semantic errors in algorithm graph schemes. It differs from the existing ones in the linear analysis time and accounting the concept of "time". The application of this method to the design workflows and software algorithms analysis will reveal complex errors in computer programs at the design stage, provide an economic effect of the software development in most organizations.

Keywords: EPC · Grammar · Design workflows · Business process

1 Introduction

This work is a development of the topic of processing workflows in software development [1–5]. The software development success problem has been studied in the business process management theory and also in theoretical informatics and the CAD theory for more than 30 years. Such attention to the problem is caused by a high degree of software development terms beyond the planned time, financial and functional costs. In the mentioned above theories, the reasons have been identified and recommendations have been developed to improve the success of computer software development, however, according to The Standish Group [6], engaged in research in the field of software development success, nowadays only 40% of developments are successfully finished.

Thus, there is no formal method for analyzing the design workflows and algorithms presented based on the graphic languages, which has linear computational complexity and takes into account the concept of "clock", therefore, a new formal method that solves the above problems in the development of computer programs, providing an increase in the success of their development.

The paper consists of the following parts. Related works review on the research topic is given in Sect. 2, a new formal design workflows and algorithms analysis method is described in Sect. 3. Results of the computational experiment are presented in Sect. 4 and afterward conclusion on the gained results is given.

© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12957, pp. 182–194, 2021.
https://doi.org/10.1007/978-3-030-87013-3_14

2 Related Works

The software development success problem is of great interest to scientists, programmers, designers [7–50]. The following articles contain a description of formal methods for processing design workflows and algorithms at the conceptual level of software development.

Van der Aalst [7] analyzes BPM literature to identify trends, focusing on modeling and DesM and EnM use cases. However, there is no mention of improving the processes themselves in terms of compliance and performance. He also with a ter Hofstede, Kiepuszewski and Barros [8] prepared a foundational document on design workflow patterns, which defines the use of twenty patterns for identifying designs. The work was subsequently used in the study of design workflows based on graphical languages such as UML [9–11], BPEL4WS [12], BML [13] and BPMN [14–16].

Kindler et al. [17] and [18] describe "The Vicious Circle" paradox, which is a semantic error in the design workflows.

Rittgen [19], proposes an OR-join implementation taking into account the difference in the number of incoming links and the number of outgoing links to the OR-join.

Levmann et al. [20] introduced the concept of "dead path elimination" to the OR-join evaluation in FlowMark.

Kiepuszewski et al. [21] provided a comprehensive discussion of issues related to structured design workflows. In [22] they investigate several fundamental issues related to the design workflows outlook, including management ones, in workflow systems, including consideration of issues such as semantic correctness and the possibility of transforming semantically equivalent models of design workflows.

Wynn et al. [23] propose a general OR-join evaluation strategy. There is a set of articles about OR-join, for example [24–26].

Van der Aalst [27] discusses the use of Petri nets to describe the design workflows, including control ones, and to verify their performance. He also provides an overview [28] of how design workflows can be verified.

3 Formal Method to Analyze Design Workflows and Algorithms

The formal method is based on the temporal RVTI automaton grammar, which is well described in [1–5]. Therefore, in this paper, we describe the methodology for constructing an RVTI analyzer and present a graph diagram of design workflows and software analysis algorithm. This technique contains two phases: synthesis and analysis.

The RVTI analyzer synthesis consists of the following stages [51]:

1. Determination of the terminal alphabet of the controlled graphic language.
2. Description of labels location.
3. Revealing of the semantic differences for links having a common graphic form.
4. Construction of the quasitherms alphabet.
5. Construction of the matrix of admissible matchings for the quasi-terminal alphabet;
6. Determination of the relations over the internal memory, providing effective control of the graphic objects connectivity.

7. Construction of RVTI grammars graph using an admissible matchings matrix, i.e., a system of relations, the vertices of which are assigned the names of rule complexes, and the arcs are quasi-terms and operations on the internal memory.

It should be noted that in addition to the graphical form, the RVTI grammar can be presented in tabular and analytical ones.

The analysis phase is carried out in two stages:

1. Minimization of the RVTI grammar.
2. Elimination of non-determinism and uncertainty.

Figure 1 shows the structure of a linearly bounded automaton [52, 53] of the developed RVTI analyzer. Arrowheads allow writing and reading stacks, and rhomboids - only reading tape cells: L_1, L_2, L_3, …, L_n.

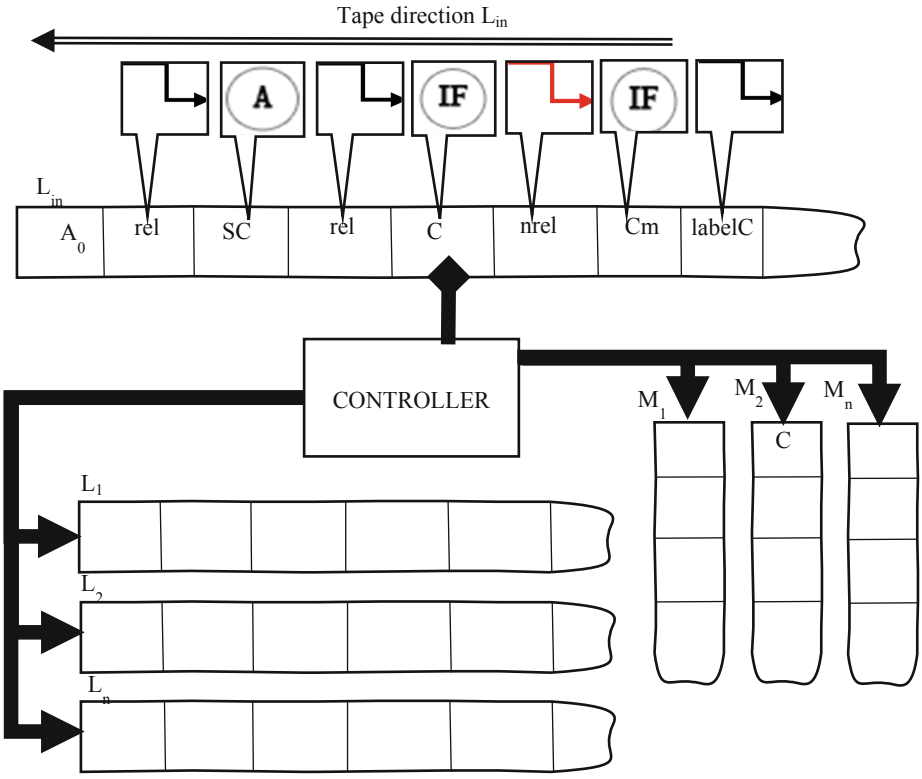

Fig. 1. Linearly bounded automaton-based RVTI analyzer structure

Design workflows and algorithms analysis method based on the graphic languages by the RVTI-analyzer is represented with the activity diagram in Fig. 2.

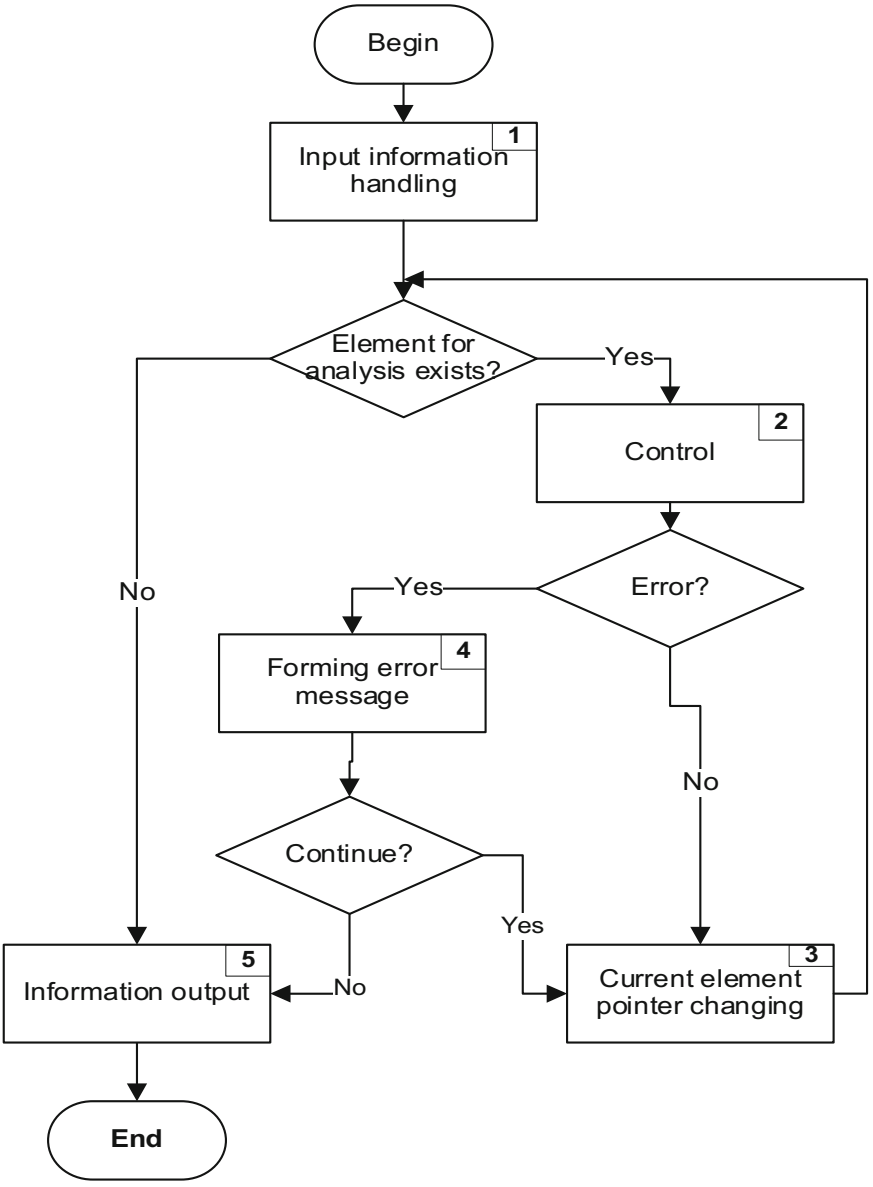

Fig. 2. Design workflows and algorithms analysis formal method

Blocks and fork items are described below.

- *Input information handling* processes the input design workflows and algorithms based on the graphic languages;
- *Element for analysis exists?* finds whether there are any unanalyzed design workflows and algorithms graphic elements;

- *Control* controls design workflows and algorithms;
- *Error?* finds whether the error is detected during the analysis;
- *Current element pointer changing* increases the counter of graphical elements;
- *Forming error message* outputs error notification;
- *Continue?* asks a designer whether to continue the analysis;
- *Information output* block displays the analysis result on the screen;
- *End* finishes the execution of the method.

The formal method has proof of linear time complexity analysis which is introduced in [4].

The list of errors detected with the authors' formal method is the following.

3. BPMN language:

   a. *Synonyms group with different parameters* – the diagram contains elements with synonymous names.
   b. *Antonyms group with same parameters* – the diagram contains elements with antonymic names.
   c. *Lack of conversion link* – elements have opposite names and there is no link between them.
   d. *No nested link to a diagram* – the diagram element is not associated with a suitable (with the same name) start element of another diagram.

4. EPC/UML language:

   a. *No start symbol* – no event found without incoming connections.
   b. *Too many outgoing links* – the event/function has more than one outgoing link.
   c. *Too many inbound links* – the event/function has more than one incoming link.
   d. *No end* – the last element is not an event/function.
   e. *Deadlock* – undecidable loop in the diagram.
   f. *Unanalyzed element* – there is an unreachable element in the diagram without incoming links.
   g. *The next figure was expected* – after the current element the next one must exist, but it does not.

5. IDEF3 language:

   a. *More than 1 initial characters* – units of behavior (UOB) with no incoming links count is greater than 1.
   b. *No start character* – no UOB without incoming links found.
   c. *Too many outcoming connections* – the UOB has more than one outcoming connection.
   d. *Too many incoming links* – the UOB has more than one incoming link.
   e. *No end* – the last element is not a UOB.
   f. *Deadlock* – an undecidable loop in a diagram.

g. *Unanalyzed element* – there is an unreachable element in the diagram without incoming links.
h. *The next figure was expected* – after the current element the next one must exist, but it does not.

6. IDEF5 language:

a. *No end* – the last element is not a class.
b. *Unanalyzed element* – there is an unreachable element in the diagram without incoming links.
c. *The next figure was expected* – after the current element the next one must exist, but it is does not.
d. *No start symbol* – class without incoming links is not found.
e. *Unknown symbol* – the diagram contains elements that do not belong to IDEF5.
f. *Many outputs* – there are elements with more than 1 outcoming link.
g. *No outputs* – there is an element without outputs.
h. *Incorrect link type* – there are links from another type of IDEF5 diagrams.
i. *Incorrect vertex type* – there are vertices from another type of IDEF5 diagrams.

4 Computational Experiment

For certain time and quantitative characteristics of the design workflows and algorithms analysis, an experiment was carried out to analyze the metrics in the EPC language basis, or with EPC diagrams. For this, various diagrams are generated and the analysis time is measured.

Target metrics include analysis time and analysis steps count. The step is meant as the number of passes along the vertex or edge of the diagram.

When generating the diagram, the following parameters are determined: the vertices count and the probability of using the EPC linear template. The diagram is constructed as a serial connection of linear and branched EPC templates. The line template of the EPC diagram is a sequential chain of events and functions (Fig. 3). The branched pattern additionally includes OR elements (Fig. 4).

The following values are used in the diagrams generation process:

- vertices count is from 100 to 100000 with a step 100;
- EPC linear diagram template usage probability is form 0 to 1 with a step of 0.2.

Additionally, the diagram has such output characteristic as the fork ratio which is counted as the ratio of edges count from condition vertices to the total edges count. The fork ratio characterizes the diagram linearity.

In summary, 1200 diagrams were built during the experiment. Based on the data obtained, various dependency graphs were built:

- the analysis time on the vertices count with forks ratio 0.02 (Fig. 5);
- the analysis steps count on vertices count (Fig. 6);

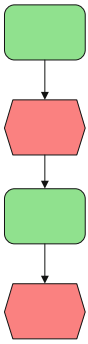

Fig. 3. Linear EPC diagram template

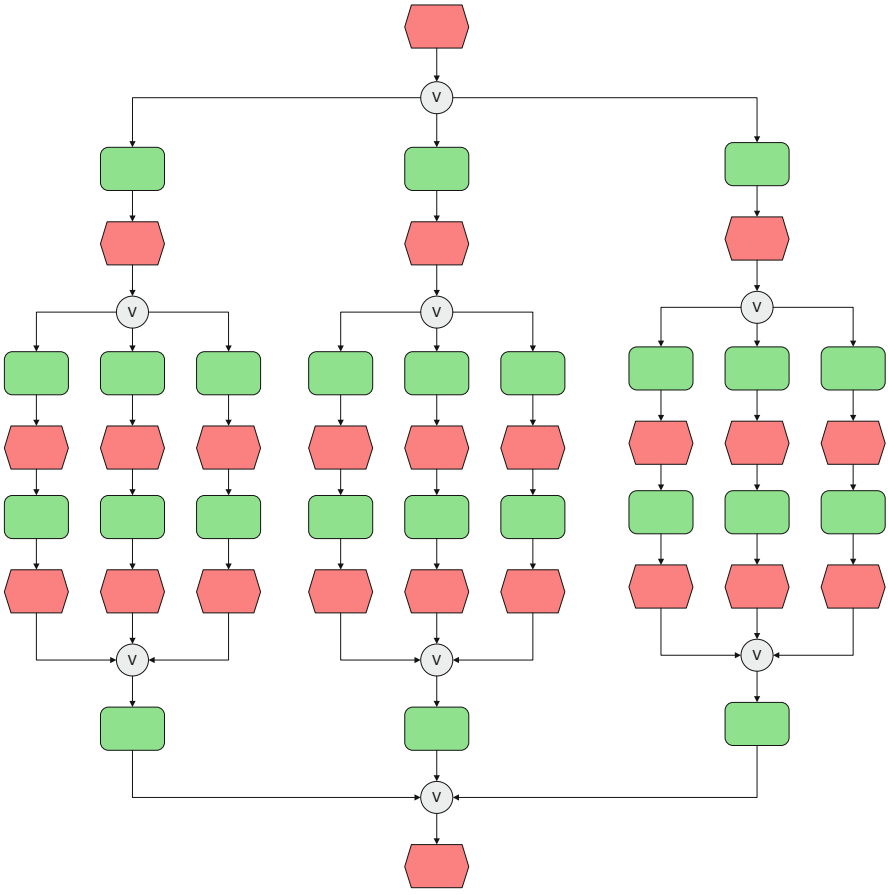

Fig. 4. Branched EPC diagram template

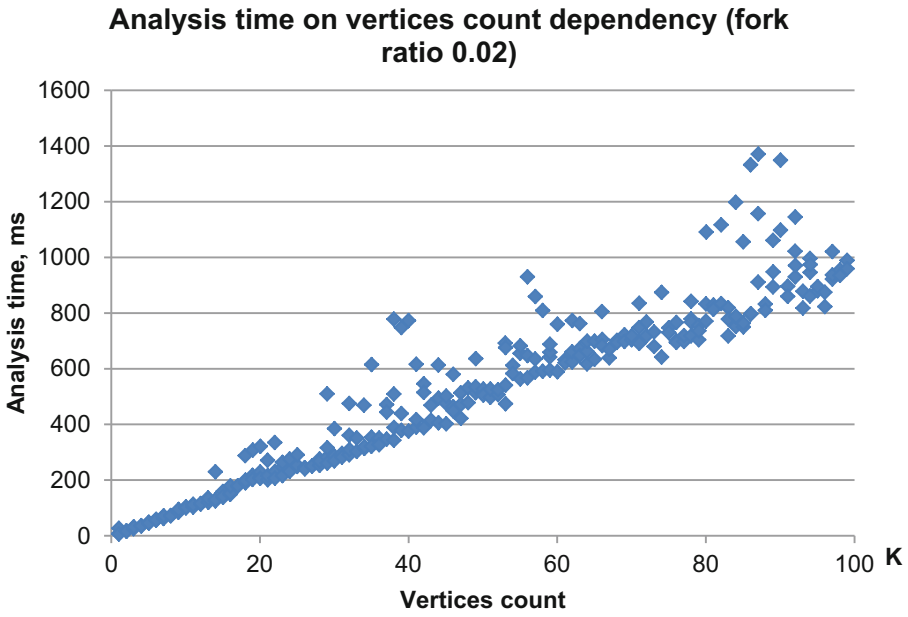

Fig. 5. Dependency graph of analysis time on the vertices count with fork ratio 0.02

- the analysis time on the elements count (Fig. 7);
- the analysis steps count on the elements count (Fig. 8).

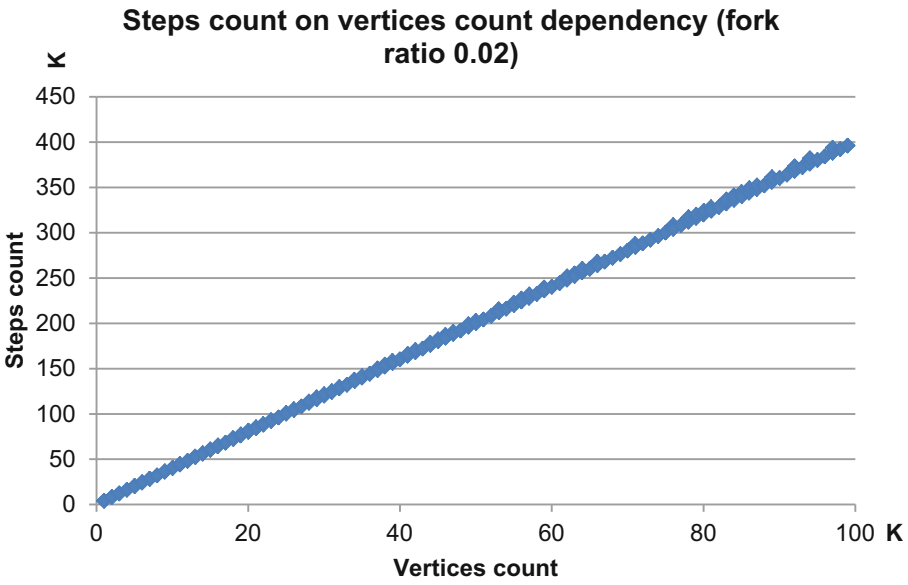

Fig. 6. Dependency graph of steps count on the vertices count with fork ratio 0.02

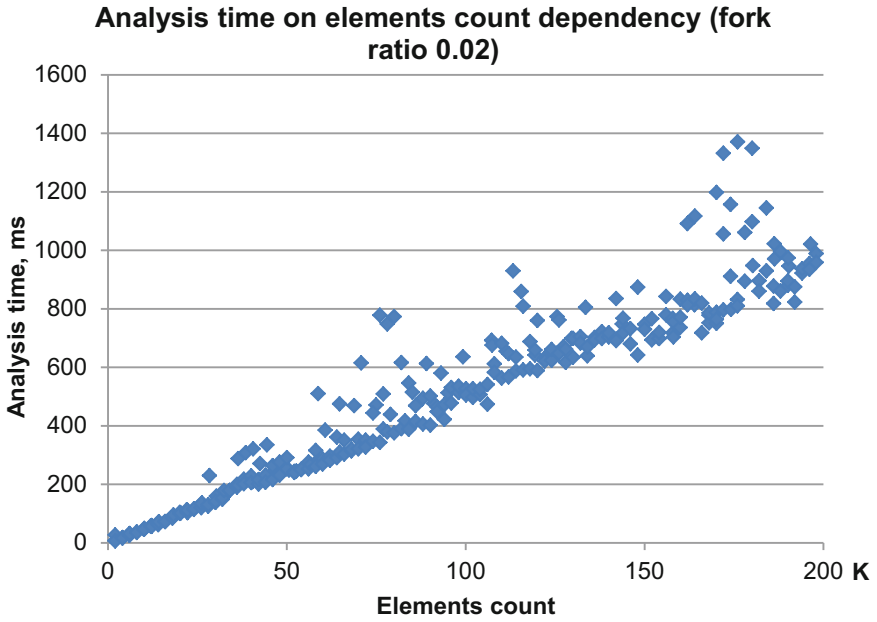

Fig. 7. Dependency graph of analysis time on the elements count with fork ratio 0.02

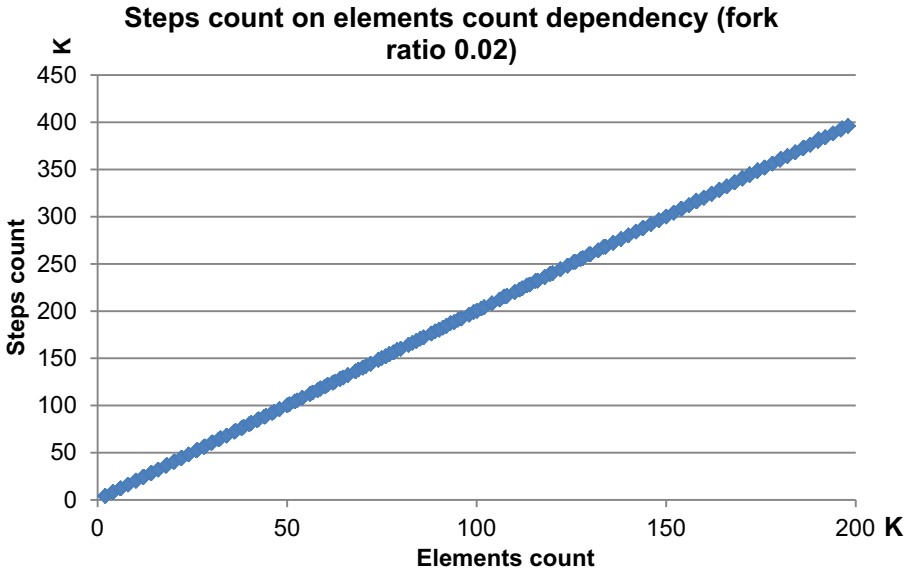

Fig. 8. Dependency graph of steps count on the elements count with fork ratio 0.02

Thus, the plotted graphs confirm the linear nature of the analysis time of the formal method.

5 Conclusion

The formal method of design workflows analysis in software development is described with the means of logic algebra and based on the temporal automaton RVTI grammar. The method differs from the existing ones in linear computational complexity in the analysis and taking into account the concept of "time". It allows the analysis of design workflows and algorithms based on various graphic languages, to identify syntactic (topological), semantic and temporal errors in design workflows and algorithms.

This novel method contributes to solving the successful software development problem. In practical terms, it allows to increase the number of detected errors at the software design stage and achieve an economic effect from its application in software development organizations.

In future works, the authors are going to draw attention to the expansion of the detected semantic errors class in the design workflows and software algorithms.

Acknowledgment. The reported research was funded by Russian Foundation for Basic Research and the government of the region of the Russian Federation, Grant no: 18-47-730032.

References

1. Afanasyev, A., Voit, N., Ukhanova, M., Ionova, I.: Development of the approach to check the correctness of workflows. Data Sci. Knowl. Eng. Sens. Decis. Support [Internet]. World Sci. https://doi.org/10.1142/9789813273238_0173. Accessed 30 July 2018
2. Afanasyev, A., Voit, N., Timofeeva, O., Epifanov, V.: Analysis and control of hybrid diagrammatical workflows. In: Abraham, A., Kovalev, S., Tarassov, V., Snasel, V., Vasileva, M., Sukhanov, A. (eds.) IITI 2017. AISC, vol. 679, pp. 124–133. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-68321-8_13
3. Afanasyev, A., Voit, N., Ukhanova, M., Ionova, I.: Analysis of design-technology workflows in the conditions of large enterprise. In: Abraham, A., Kovalev, S., Tarassov, V., Snasel, V., Vasileva, M., Sukhanov, A. (eds.) IITI 2017. AISC, vol. 679, pp. 134–140. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-68321-8_14
4. Voit, N., Bochkov, S., Kirillov, S.: Temporal automaton RVTI-grammar for the diagrammatic design workflow models analysis. In: 2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT) [Internet]. IEEE, 7 October 2020. https://doi.org/10.1109/aict50176.2020.9368810
5. Afanasyev, A.N., Voit, N.N., Kirillov, S.Y.: Temporal automata RVTI-grammar for processing diagrams in visual languages as BPMN, eEPC and Askon-Volga. In: Proceedings of the 2019 5th International Conference on Computer and Technology Applications [Internet]. ACM, 16 Apr 2019. https://doi.org/10.1145/3323933.3324067
6. The Standish Group. https://www.standishgroup.com/
7. van der Aalst, W.M.P.: Business process management: a comprehensive survey. ISRN Software Engineering, pp. 1–37 (2013)
8. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distrib. Parallel Databases **14**(3), 5–51 (2003)
9. Dumas, M., ter Hofstede, A.H.M.: UML activity diagrams as a workflow specification language. In: Gogolla, M., Kobryn, C. (eds.) UML 2001. LNCS, vol. 2185, pp. 76–90. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45441-1_7

10. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Wohed, P.: On the suitability of UML 2.0 activity diagrams for business process modelling. In: Stumptner, M., Hartmann, S., Kiyoki, Y. (eds.) Proceedings of the Third Asia-Pacific Conference on Conceptual Modelling (APCCM2006). CRPIT, vol. 53, pp. 95–104. ACS, Hobart (2006)
11. Wohed, P., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M., Russell, N.: Pattern-based analysis of the control-flow perspective of UML activity diagrams. In: Delcambre, L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, O. (eds.) ER 2005. LNCS, vol. 3716, pp. 63–78. Springer, Heidelberg (2005). https://doi.org/10.1007/11568322_5
12. Wohed, P., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M.: Analysis of web services composition languages: the case of BPEL4WS. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) ER 2003. LNCS, vol. 2813, pp. 200–215. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39648-2_18
13. Wohed, P., Perjons, E., Dumas, M., ter Hofstede, A.H.M.: Pattern based analysis of EAI languages—the case of the business modeling language. In: Camp, O., Piattini, M. (eds.) Proceedings of the 5th International Conference on Enterprise Information Systems (ICEIS 2003), vol. 3, pp. 174–184. Escola Superior de Tecnologia do Instituto Politécnico de Setúbal, Angers (2003)
14. OMG/BPMI: Business Process Model and Notation (BPMN) Version 2.0.2 (2014). http://www.omg.org/spec/BPMN/2.0.2/
15. White, S.: Process modeling notations and workflow patterns. In: Fischer, L. (ed.) Workflow Handbook 2004, pp. 265–294. Future Strategies Inc., Lighthouse Point (2004)
16. Wohed, P., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M., Russell, N.: On the suitability of BPMN for business process modelling. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 161–176. Springer, Heidelberg (2006). https://doi.org/10.1007/11841760_12
17. van der Aalst, W.M.P., Desel, J., Kindler, E.: On the semantics of EPCs: a vicious circle. In: Rump, M., Nüttgens, F.J. (eds.) Proceedings of the EPK 2002: Business Process Management Using EPCs, pp. 71–80. Gesellschaft fur Informatik, Trier (2002)
18. Kindler, E.: On the semantics of EPCs: resolving the vicious circle. Data Knowl. Eng. **56**(1), 23–40 (2006)
19. Rittgen, P.: From process model to electronic business process. In: Avison, D., Christiaanse, E., Ciborra, C.U., Kautz, K., Pries-Heje, J., Valor, J. (eds.) Proceedings of the European Conference on Information Systems (ECIS 1999), Copenhagen, Denmark, pp. 616–625 (1999). http://www.adm.hb.se/~pri/ecis99.pdf
20. Leymann, F., Roller, D.: Production Workflow: Concepts and Techniques. Prentice Hall, Upper Saddle River (2000)
21. Kiepuszewski, B., ter Hofstede, A.H.M., Bussler, C.J.: On structured workflow modelling. In: Wangler, B., Bergman, L. (eds.) CAiSE 2000. LNCS, vol. 1789, pp. 431–445. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45140-4_29
22. Kiepuszewski, B., ter Hofstede, A.H.M., van der Aalst, W.M.P.: Fundamentals of control flow in workflows. Acta Informatica **39**(3), 143–209 (2003)
23. Wynn, M.T., Edmond, D., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Achieving a general, formal and decidable approach to the OR-join in workflow using reset nets. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 423–443. Springer, Heidelberg (2005). https://doi.org/10.1007/11494744_24
24. Mendling, J., van der Aalst, W.: Formalization and verification of EPCs with or-joins based on state and context. In: Krogstie, J., Opdahl, A., Sindre, G. (eds.) CAiSE 2007. LNCS, vol. 4495, pp. 439–453. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72988-4_31

25. Dumas, M., Grosskopf, A., Hettel, T., Wynn, M.: Semantics of standard process models with OR-joins. In: Meersman, R., Tari, Z. (eds.) OTM 2007. LNCS, vol. 4803, pp. 41–58. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76848-7_5
26. Völzer, H.: A new semantics for the inclusive converging gateway in safe processes. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 294–309. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15618-2_21
27. van der Aalst, W.M.P.: The application of Petri nets to workflow management. J. Circ. Syst. Comput. **8**(1), 21–66 (1998)
28. van der Aalst, W.M.P., van Hee, K.M., ter Hofstede, A.H.M., Verbeek, H.M.W., Voorhoeve, M., Wynn, M.T.: Soundness of workflow nets: classification, decidability, and analysis. Formal Aspects Comput. **23**(3), 333–363 (2011)
29. Simsion, G., Witt, G.: Data Modeling Essentials, 3rd edn. Elsevier, San Francisco (2005)
30. Hay, D.C.: Data Model Patterns: Conventions of Thought. Dorset House, Wiley, New York (2001)
31. Gray, J., Reuter, A.: Transaction Processing: Concepts and Techniques. Morgan Kaufmann Publishers, San Francisco (1992)
32. Börger, E.: Approaches to modeling business processes. A critical analysis of BPMN, workflow patterns and YAWL. Softw. Syst. Model. **11**(3), 305–318 (2012)
33. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. The MIT Press, Cambridge (1999)
34. Eriksson, H.E., Penker, M.: Business Modeling with UML. OMG Press, New York (2000)
35. Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M.H., La Rosa, M.: Configurable workflow models. Int. J. Coop. Inf. Syst. **17**(2), 177–221 (2008)
36. Hagen, C., Alonso, G.: Exception handling in workflow management systems. IEEE Trans. Softw. Eng. **26**(10), 943–958 (2000)
37. ter Hofstede, A.H.M., van der Aalst, W.M.P., Adams, M., Russell, N. (eds.): Modern Business Process Automation: YAWL and Its Support Environment. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03121-2
38. Holt, A.W.: Coordination technology and Petri nets. In: Rozenberg, G. (ed.) APN 1985. LNCS, vol. 222, pp. 278–296. Springer, Heidelberg (1986). https://doi.org/10.1007/BFb0016217
39. Huang, Y.N., Shan, M.C.: Policies in a resource manager of workflow systems: modeling, enforcement and management. Technical report HPL-98–156, Hewlett-Packard Company (1999). http://www.hpl.hp.com/techreports/98/HPL-98-156.pdf
40. Jablonski, S., Bussler, C.: Workflow Management: Modeling Concepts, Architecture and Implementation. Thomson Computer Press, London (1996)
41. Kiepuszewski, B.: Expressiveness and suitability of languages for control flow modelling in workflows. Ph.D. thesis, Queensland University of Technology, Brisbane, Australia (2003)
42. Marshall, C.: Enterprise Modeling with UML. Addison Wesley, Reading (1999)
43. zur Muehlen, M.: Workflow-Based Process Controlling: Foundation, Design and Application of Workflow Driven Process Information Systems. Logos, Berlin (2004)
44. OMG: OMG Unified Modeling Language (OMG UML), Version 2.5 (2015). http://www.omg.org/spec/UML/2.5
45. Russell, N., van der Aalst, W., ter Hofstede, A.: Workflow exception patterns. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 288–302. Springer, Heidelberg (2006). https://doi.org/10.1007/11767138_20
46. Sadiq, S., Orlowska, M., Sadiq, W., Foulger, C.: Data flow and validation in workflow modelling. In: Schewe, K.D., Williams, H.E. (eds.) Proceedings of the 5th Australasian Database Conference (ADC 2004). CRPIT, vol. 27, pp. 207–214. ACS, Dunedin (2004)
47. Verbeek, H.M.W., Basten, T., van der Aalst, W.M.P.: Diagnosing workflow processes using Woflan. Comput. J. **44**(4), 246–279 (2001)

48. Glushkov, V.M., Zeitlin, G.E., Yuschenko, E.L.: Algebra. Programming Languages, 2rd edn, 320 p. Naukova dumka, Kiev, USSR (1978)
49. Tonbul, G., et al.: Error density metrics for business process model. In: 2009 24th International Symposium on Computer and Information Sciences, pp. 513–517. IEEE (2009)
50. Misra, S., et al.: A proposed pragmatic software development process model. In: Intelligent Systems: Concepts, Methodologies, Tools, and Applications, pp. 448–462 (2018)
51. Afanasyev, A.N., Voit, N.N.: Structural-semantic workflows analysis and designers training in design automation. UlSTU, Ulyanovsk (2018). http://venec.ulstu.ru/lib/disk/2017/467.pdf
52. Myhill, J.: Linear Bounded Automata (WADD Technical Note). Wright Patterson AFB, Wright Air Development Division, Ohio (1960)
53. Kuroda, S.-Y.: Classes of languages and linear-bounded automata. Inf. Contr. **7**(2), 207–223 (1964)